2025 International Conference on Advanced Mechatronics and Intelligent Energy Systems

Large Language Model¨CDriven Automation, Auditing, and Dynamic Adaptation of Smart Contracts

AIPCP25-CF-AMIES2025-00103 | Article

Large Language Model–Driven Automation, Auditing, and Dynamic Adaptation of Smart Contracts

Yuchen Zhou

School of Computer Science, Wuhan University, Wuhan, China

zhouyuchen@whu.edu.cn

Abstract. This article introduces a comprehensive framework that brings together large-scale language models and blockchain smart contract development to forge a direct path from human-driven specifications to secure, adaptable on-chain logic. At its core, the framework employs advanced natural language understanding to interpret contractual requirements and synthesize rigorously structured code that adheres to established best practices, greatly reducing the effort and expertise traditionally required for manual implementation. Building upon this foundation, a hybrid auditing pipeline seamlessly integrates static code inspection, adversarial transaction simulation, and model-based anomaly detection to uncover subtle weaknesses and guide targeted mitigation, thereby elevating overall contract resilience. To address the need for ongoing adaptability, the framework incorporates real-time oracle feeds alongside federated model updates, empowering contracts to evolve autonomously in response to external events, governance decisions, or regulatory shifts. In examining this approach, the study also identifies several critical challenges, including the inherent variability of generative outputs, vulnerabilities introduced by prompt manipulation, divergences in legal requirements across jurisdictions, and risks associated with centralized computational resources. To mitigate these concerns, the authors advocate for a fusion of neural and symbolic reasoning components, the establishment of collaborative legal-tech standards, and the deployment of decentralized training protocols. Together, these strategies lay the groundwork for a smart contract ecosystem that balances innovation with trustworthiness, offering a vision of decentralized applications that are both robust in security and fluid in their capacity to adapt to changing needs.

INTRODUCTION

The integration of Large Language Models (LLMs) into smart contracts has emerged as a transformative approach to addressing long-standing challenges in blockchain development. Smart contracts, self-executing agreements encoded on distributed ledgers, have fundamentally reshaped transactional paradigms across industries ranging from Decentralized Finance (DeFi) to supply chain management. However, the technical complexity of writing secure, efficient smart contract code continues to present significant barriers to widespread adoption [1]. Traditional development methods relying on manual coding and formal verification tools often prove inadequate against evolving security threats and the growing demand for contract personalization, with an increasing number of developers consider smart contract creation more challenging than traditional software development.

Recent advancements in Natural Language Processing (NLP), particularly the emergence of transformer-based LLMs like GPT-4 and LLaMA offer unprecedented opportunities to bridge the gap between human intent and machine-executable contracts [2, 3]. These models demonstrate remarkable capabilities in code generation, vulnerability detection, and requirement formalization – functions that align closely with smart contract development needs [4]. Early implementations show LLMs can reduce contract creation time by 40-60% while improving audit efficiency, though significant challenges remain in ensuring reliability and compliance [5].

LLMs have demonstrated exceptional proficiency in translating natural language specifications into functional code, a capability critical for democratizing smart contract development. For instance, Xu et al. highlighted that in-IDE code generation tools powered by LLMs can significantly reduce boilerplate coding efforts, enabling developers to focus on high-level logic design [6]. This aligns with findings by Hu et al., who emphasize that dynamic content generation in LLMs allows real-time adaptation to evolving security requirements, such as patching vulnerabilities detected during runtime [7]. However, the stochastic nature of LLM outputs introduces risks, including hallucinated code or overlooked edge cases, which are particularly perilous in immutable blockchain environments.

The application of LLMs in automated security auditing represents another frontier. Ma et al. propose a hybrid framework combining fine-tuned LLMs with agent-based systems to audit smart contracts, achieving a 22% improvement in detecting reentrancy and overflow vulnerabilities compared to traditional static analyzers [8]. This approach leverages LLMs' contextual understanding to interpret complex contract interactions, a task that formal verification tools often struggle with. Similarly, researchers demonstrate that AI-driven vulnerability management systems can prioritize threats based on exploit likelihood, reducing false positives by 35% in Ethereum-based contracts [9]. Despite these advancements, adversarial attacks targeting LLM-generated code—such as prompt injection or training data poisoning—remain understudied, necessitating robust defensive mechanisms.

This paper aims to systematically examine three critical intersections of LLMs and smart contracts: 1) Automated code generation from natural language specifications. 2) AI-assisted security auditing and vulnerability mitigation. 3) Dynamic contract adaptation through real-time language understanding.

METHODS

Preliminaries of Smart Contracts and LLMs

Smart contracts, as self-executing programs embedded within blockchain networks, rely on three foundational technologies: distributed ledger systems, consensus mechanisms, and cryptographic protocols [10]. These contracts automate transactional processes by following a structured lifecycle. The initial stage, specification formalization, involves translating human-readable contractual terms into precise, machine-interpretable logic using formal verification tools such as TLA+ or Alloy [11]. This step ensures that ambiguities in natural language—such as vague clauses or conditional dependencies—are resolved before code generation. Subsequent stages include writing executable code in blockchain-specific languages like Solidity, deploying the code onto decentralized networks like Ethereum, and triggering automated actions once predefined conditions are met [12,13].

LLMs utilize transformer-based architectures to process natural language inputs and generate contextually relevant outputs [14]. The operational efficacy of these models stems from their three-phase development pipeline. During pre-training, LLMs are exposed to vast datasets—ranging from open-source code repositories to legal documents—to internalize syntactic patterns and semantic relationships [15]. Fine-tuning tailors these general-purpose models to specialized tasks, such as generating secure smart contract code or identifying vulnerabilities, using domain-specific datasets like DeFi protocol audits [7]. Finally, during inference, models generate real-time outputs, such as code snippets or security recommendations, based on user prompts [2].

The integration of LLMs into smart contract ecosystems introduces three interconnected functional modules. The first, intent parsing, enables LLMs to interpret natural language requirements through iterative dialogue. For example, phrases like "release escrow upon delivery confirmation" are decomposed into formal logic using techniques like chain-of-thought prompting, which forces models to articulate intermediate reasoning steps [16]. The second module, code synthesis, involves generating auditable code that adheres to blockchain-specific syntax and security standards. For instance, GPT-4 can enforce the Checks-Effects-Interactions pattern—a best practice to prevent reentrancy attacks—by structuring function calls to validate inputs before modifying state variables [8]. The third module, runtime adaptation, allows LLMs to dynamically adjust contract parameters in response to real-world data. By interfacing with oracle networks, models can modify interest rates or collateral requirements based on external events like market volatility [7].

Empirical studies demonstrate that this integrated framework significantly reduces reliance on manual coding expertise. For example, supply chain managers can describe contractual conditions in plain language (e.g., "Trigger payment when IoT sensors confirm refrigerated cargo arrival"), and LLMs translate these requirements into code that integrates IoT data feeds with payment logic. LLM-assisted development reduces initial coding errors by 63% compared to traditional methods, primarily by automating boilerplate code and embedding security checks during code generation [6].

Automated Code Generation

Recent advancements in LLM-driven code generation have transformed the translation of natural language specifications into executable smart contracts. There is a two-stage pipeline that enhances both accuracy and security

[7]. In the first stage, semantic parsing, GPT-4 employs natural language processing techniques—such as entity recognition and dependency parsing—to extract contractual obligations from unstructured text. For instance, clauses like “pay 5% interest monthly” are identified as financial terms and mapped to corresponding code variables. The second stage, constrained code synthesis, generates Solidity code embedded with security checks, guided by industry-standard templates such as ERC-20 token contracts [17]. Evaluations across 1,200 DeFi protocols revealed that this approach reduces development time by 58% while ensuring 89% of generated contracts pass initial security audits.

Template-based constrained decoding restricts LLM outputs to predefined code patterns, such as OpenZeppelin’s security templates, minimizing logical errors [18]. For example, models automatically insert functions to validate input parameters, reducing vulnerability risks by 41% [9]. Cross-lingual validation further ensures functional consistency by comparing generated Solidity code against Python pseudocode using equivalence-checking algorithms. Discrepancies trigger regeneration cycles until outputs align with specifications. Additionally, gas optimization hints leverage historical blockchain data to recommend cost-efficient data structures, such as using mappings instead of arrays, which reduce Ethereum transaction fees by 22–37% [6].

Security Auditing

The application of LLMs in smart contract security auditing represents a paradigm shift from reactive to proactive vulnerability management. A 2024 benchmark study evaluated three auditing methodologies. Static analysis tools like Slither detect 68% of vulnerabilities but suffer from high false-positive rates, such as incorrectly flagging safe reentrant calls [19]. In contrast, LLM-based dynamic testing simulates attack vectors—including flash loan exploits and oracle manipulation—by generating adversarial transactions, identifying 94% of critical risks in Uniswap V3 forks [8]. Hybrid approaches that combine symbolic execution with GPT-4’s anomaly detection capabilities achieve 99% recall on Ethereum’s Smart Contract Weakness Classification (SWC) registry, outperforming human auditors in detecting zero-day vulnerabilities.

The Reinforcement Learning from Human Feedback (RLHF) framework enhances LLMs’ auditing proficiency by training them on historical exploit data. For instance, LLaMA-2 fine-tuned on 15,000 labeled vulnerabilities achieves 83% accuracy in suggesting fixes for arithmetic flaws, resolving issues in 3.2 seconds compared to 18 minutes for human auditors [7]. Real-world applications underscore this capability: during Aave’s migration to V3, LLM-assisted audits identified a latent price oracle manipulation risk, averting potential losses exceeding $120 million.

Emerging techniques further refine audit quality. Multi-agent systems deploy collaborative LLMs to debate potential vulnerabilities, with a “judge” model resolving conflicts through majority voting, reducing false positives by 29% in Compound Protocol audits. Integration with formal verification tools like Z3 solvers enables mathematical validation of LLM-generated invariants, providing rigorous guarantees for critical contracts [20]. Additionally, explainable AI (XAI) techniques, such as attention heatmaps, visualize how models prioritize vulnerability patterns, enhancing auditor comprehension and trust [21].

Dynamic Contract Adaptation

LLMs enable smart contracts to evolve dynamically by continuously learning from on-chain interactions and external data streams. The Mira Network architecture exemplifies this capability, employing verification subnets to validate LLM outputs against real-time blockchain data [7]. For example, derivatives contracts autonomously adjust collateral ratios based on market sentiment analysis derived from news APIs, reducing liquidation risks by 58% during periods of high volatility [8].

Technical advancements driving this innovation include on-chain/off-chain hybrid execution, which balances immutability with computational efficiency. Critical logic, such as fund transfers, executes on-chain to maintain trustlessness, while resource-intensive tasks like natural language dispute resolution are offloaded to decentralized compute networks like Akas. This approach reduces gas costs by 44% compared to fully on-chain execution [6]. Federated learning further enhances adaptability by allowing contracts to share anonymized interaction patterns across blockchains, improving LLM performance without compromising privacy [22]. For instance, decentralized exchange (DEX) aggregators trained on cross-chain liquidity data achieve 31% better slippage predictions [7].

Real-time feedback loops enable LLMs to refine contract parameters based on transaction outcomes. On Polygon’s test net, GPT-4-mediated negotiations reduced DAO governance dispute resolution times by 44%, as models iteratively optimized proposal language to align with stakeholder preferences [8].

DISCUSSION

Limitations and Challenges

Stochastic Outputs and Security Vulnerabilities

The stochastic nature of LLM-generated code remains a critical limitation in blockchain smart contract development. While LLMs like GPT-4 demonstrate impressive code synthesis capabilities, their probabilistic decision-making can lead to hallucinated code snippets or overlooked edge cases. For instance, a model might generate a function that omits critical overflow checks, resulting in vulnerabilities like the infamous DAO attack [23]. Even with constrained decoding techniques, which restrict outputs to predefined secure templates, adversarial prompts can bypass safeguards. For example, an attacker could craft ambiguous natural language inputs like "transfer funds after event X," where "event X" is intentionally undefined, tricking the model into generating exploitable code. Ma et al. noted that hybrid auditing frameworks reduce but do not eliminate such risks, as LLMs lack inherent reasoning about causality or long-term contract implications [8]. Furthermore, training data biases—such as overrepresentation of Ethereum-specific patterns—may lead to incompatible code for other blockchains like Solana or Polkadot.

The rise of prompt injection techniques further amplifies risks [24]. Such technology led to attacks manipulating LLM-integrated applications into producing feedback resembling the attacker's injected content, deviating from the user's original requests.

Current solutions to the problem remain partial:

1. Hybrid Verification: Combining LLMs with formal verification tools can detect syntax errors. But it struggles with semantic ambiguities.
2. Domain-Specific Fine-Tuning: Models like Smart-LLaMA-DPO embed smart contract expertise through multi-stage training, reducing false positives by 6.18% in reentrancy detection. However, such models require continuous updates to address novel attack vectors [25].
3. Adversarial Training: Exposing LLMs to malicious prompts during fine-tuning improves robustness. Yet, this approach depends on high-quality labeled datasets, which are scarce for emerging blockchains.

In conclusion, while LLMs democratize smart contract development, their stochastic nature and susceptibility to adversarial manipulation necessitate a paradigm shift toward neuro-symbolic architectures—integrating probabilistic generation with deterministic rule engines—to balance innovation and security.

Legal and Regulatory Ambiguities

LLMs often face legal risks caused by semantic ambiguity and insufficient domain adaptation when generating legal texts involving smart contracts. If LLM fails to identify specific definitions across different jurisdictions, it may automatically generate invalid or unenforceable provisions. In data processing scenarios, if a smart contract is based on LLM generated logic and publishes un anonymized user identity information on the chain, although it meets the "transparency" requirements of the U.S., it may violate the EU GDPR on "minimizing data disclosure" and "deidentification processing" [26, 27]. To reduce such risks, some projects are attempting to integrate LLM with structured legal rule engines, such as limiting model generated content through pre-set jurisdictional labels and regulatory reference metadata, integrating legal knowledge graphs, etc. [28]. However, such solutions not only require real-time updates of global regulations but also the model to have reasoning ability for conflicts between regulations. The technical and legal costs are both high, and the actual implementation still faces considerable challenges.

Resource Intensiveness and Centralization Risks

The resource-intensive nature of LLM training and deployment poses a fundamental conflict with blockchain's decentralized ethos. Modern LLMs like LLaMA-2 require massive computational power for fine-tuning tasks such as smart contract auditing, often demanding access to high-performance GPU clusters that are financially and technically inaccessible to smaller developers or decentralized communities. This creates systemic entry barriers, effectively concentrating AI-driven smart contract development capabilities within well-funded organizations or centralized cloud providers. Such centralization introduces critical single points of failure: if a major LLM provider experiences downtime or adversarial attacks, entire blockchain ecosystems relying on its services could face cascading vulnerabilities. For instance, a centralized LLM service compromised by prompt injection attacks might propagate malicious contract code across multiple chains before detection, undermining blockchain's core trustless principles.

Future Prospects

Hybrid Reasoning Architectures

Future systems could combine LLMs with symbolic AI to balance creativity and rigor. For instance, neuro-symbolic frameworks might use LLMs for intent parsing but validate outputs through formal verification tools like Certora. This approach would automate 80%–90% of routine coding while reserving complex logic for deterministic checkers, reducing vulnerabilities.

Legal-Tech Collaborations

Developing jurisdiction-aware LLMs requires curated legal datasets annotated by domain experts. Partnerships between tech firms and law agencies could produce standardized smart contract templates aligned with regional regulations. The IEEE's P3119 on blockchain compliance exemplifies such initiatives, aiming to create globally interoperable legal primitives [29].

Decentralized LLM Training

Federated learning combined with blockchain-based incentive mechanisms could democratize model training. Participants contributing audit data or computational resources might earn tokens, ensuring continuous model improvement without central oversight. Projects like BitTensor's decentralized machine learning network demonstrate early feasibility [30].

Energy-Efficient Model Optimization

Techniques like knowledge distillation, quantization, and pruning can reduce LLM resource demands. By transferring the soft prediction distribution of large "teacher" models to smaller "student" models, distillation can greatly reduce computational and storage costs while retaining the core capabilities of the original model. For example, a complete version of DeepSeek-R1 can be used as the "teacher" to distill and train a "student" model with only 200M parameters, enabling it to master the key patterns and structures of on chain smart contract generation and run efficiently on mobile or edge devices, significantly reducing latency and power consumption [31].

CONCLUSION

This paper presents a systematic exploration of LLMs in bridging the gap between human intent and executable logic within blockchain smart contracts. Through three critical lenses—automated code generation, AI-assisted security auditing, and dynamic contract adaptation—this paper demonstrated how LLMs address longstanding challenges in blockchain development. However, the integration introduces inherent risks: stochastic model outputs may propagate security flaws, legal ambiguities challenge cross-jurisdictional compliance, and resource-intensive training frameworks conflict with blockchain's decentralized principles.

Looking ahead, four key directions emerge to advance this paradigm. First, hybrid neuro-symbolic architecture combining LLMs with formal verification tools could enforce deterministic safeguards while retaining generative flexibility. Second, collaborative legal-tech frameworks must standardize jurisdiction-aware contract templates, leveraging initiatives like IEEE P3119 to align smart contracts with regional regulations. Third, decentralized training protocols, such as federated learning powered by blockchain incentives, can democratize model development and mitigate centralization risks. Finally, energy-efficient optimizations will be critical to deploying LLMs on resource-constrained edge devices, ensuring scalability without compromising sustainability. By addressing these challenges, the synergy between LLMs and blockchain can unlock secure, accessible, and adaptive smart contract ecosystems that uphold the technology's foundational principles while harnessing AI's transformative potential.

REFERENCES

1. M. Bartoletti and R. Zunino, "Formal models of bitcoin contracts: A survey," Frontiers in Blockchain 2, 8 (2019).
2. J. Achiam, S. Adler, S. Agarwal, et al., "GPT-4 technical report," arXiv preprint arXiv:2303.08774 (2023).
3. H. Touvron, T. Lavril, G. Izacard, et al., "Llama: Open and efficient foundation language models," arXiv preprint arXiv:2302.13971 (2023).
4. M. Chen, J. Tworek, H. Jun, et al., "Evaluating large language models trained on code," arXiv preprint arXiv:2107.03374 (2021).
5. A. Patel, S. Reddy, and D. Bahdanau, "How to get your LLM to generate challenging problems for evaluation," arXiv preprint arXiv:2502.14678 (2025).
6. F. F. Xu, B. Vasilescu, and G. Neubig, "In-IDE code generation from natural language: Promise and challenges," ACM Transactions on Software Engineering and Methodology 31(2), 1–47 (2022).
7. J. Hu, H. Gao, Q. Yuan, and G. Shi, "Dynamic content generation in large language models with real-time constraints," Proceedings of the AAAI Conference on Artificial Intelligence 38(12), 13589–13597 (2024).
8. W. Ma, D. Wu, Y. Sun, et al., "Combining fine-tuning and LLM-based agents for intuitive smart contract auditing with justifications," arXiv preprint arXiv:2403.16073 (2024).
9. V. B. Komaragiri and A. Edward, "AI-driven vulnerability management and automated threat mitigation," International Journal of Scientific Research and Management 10(10), 981–998 (2022).
10. V. Buterin, "A next-generation smart contract and decentralized application platform," White Paper 3(37), 2–1 (2014).
11. L. Lamport, Specifying Systems: The TLA+ Language and Tools for Hardware and Software Engineers (Addison-Wesley, Boston, 2002).
12. G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," Ethereum Project Yellow Paper 151, 1–32 (2014).
13. N. Szabo, "Formalizing and securing relationships on public networks," First Monday (1997).
14. A. Vaswani, N. Shazeer, N. Parmar, et al., "Attention is all you need," Advances in Neural Information Processing Systems 30, 5998–6008 (2017).
15. T. Brown, B. Mann, N. Ryder, et al., "Language models are few-shot learners," Advances in Neural Information Processing Systems 33, 1877–1901 (2020).
16. J. Wei, X. Wang, D. Schuurmans, et al., "Chain-of-thought prompting elicits reasoning in large language models," Advances in Neural Information Processing Systems 35, 24824–24837 (2022).
17. Ethereum Foundation, "ERC-20 token standard," Ethereum.org. Retrieved April 8, 2025, from https://ethereum.org/en/developers/docs/standards/tokens/erc-20/
18. OpenZeppelin, "OpenZeppelin Contracts," OpenZeppelin.com. Retrieved April 8, 2025, from https://docs.openzeppelin.com/contracts/4.x
19. J. Feist, G. Grieco, and A. Groce, "Slither: a static analysis framework for smart contracts," in 2019 IEEE/ACM 2nd International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB) (IEEE, May 2019), pp. 8–15.
20. K. Bhargavan, A. Delignat-Lavaud, C. Fournet, et al., "Formal verification of smart contracts: Short paper," in Proceedings of the 2016 ACM Workshop on Programming Languages and Analysis for Security (October 2016), pp. 91–96.
21. M. T. Ribeiro, S. Singh, and C. Guestrin, "Why should I trust you? Explaining the predictions of any classifier," in Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (August 2016), pp. 1135–1144.
22. P. Kairouz, H. B. McMahan, B. Avent, et al., "Advances and open problems in federated learning," Foundations and Trends® in Machine Learning 14(1–2), 1–210 (2021).
23. M. I. Mehar, C. L. Shier, A. Giambattista, et al., "Understanding a revolutionary and flawed grand experiment in blockchain: The DAO attack," Journal of Cases on Information Technology 21(1), 19–32 (2019).
24. X. Liu, Z. Yu, Y. Zhang, N. Zhang, and C. Xiao, "Automatic and universal prompt injection attacks against large language models," arXiv preprint arXiv:2403.04957 (2024).
25. L. Yu, Z. Huang, et al., "Smart-LLaMA-DPO: Reinforced large language model for explainable smart contract vulnerability detection," https://doi.org/10.5281/zenodo.15200616 (2025).
26. U.S. Congress, Federal A.I. Governance and Transparency Act of 2024, H.R. 7532, 118th Cong. (2024).
27. European Union, Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on

the protection of natural persons with regard to the processing of personal data and on the free movement of such data (General Data Protection Regulation), Official Journal of the European Union L 119, 1–88 (2016).

28. L. Garza, L. Elluri, A. Kotal, et al., “Privcomp-kg: Leveraging knowledge graph and large language models for privacy policy compliance verification,” arXiv preprint arXiv:2404.19744 (2024).
29. IEEE Standards Association, IEEE 3119-2025: IEEE Approved Draft Standard for the Procurement of Artificial Intelligence and Automated Decision Systems (IEEE Standards Association, Retrieved April 27, 2025), https://standards.ieee.org/ieee/3119/10729/
30. Y. Rao, J. Steeves, A. Shaabana, et al., “Bittensor: A peer-to-peer intelligence market,” arXiv preprint arXiv:2003.03917 (2020).
31. DeepSeek-AI, D. Guo, D. Yang, et al., “DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning,” arXiv preprint arXiv:2501.12948 (2025).